AmigaTalk

COLLABORATORS			
	TITLE : AmigaTalk		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		July 10, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

AmigaTalk

1.1 AmigaTalk© 1998-2000 User Manual:

This manual is intended as an aid in using the AmigaTalk©
system. It is not intended to be used as an introduction to
the SmallTalk language. AmigaTalk© is largely (with exceptions listed
in a later section) a subset of the SmallTalk-80* language
described in SmallTalk blue
A complete description of the classes included in the AmigaTalk system
and the messages they accept is given in Appendix 1
AmigaTalk was generated from Little Smalltalk V1.0, written by
Timothy Budd. I know that there are more modern versions of Little
Smalltalk in existence; however, they load & save system image files
in a binary format that is not documented.
Requirements What type of system do I need?
File System Configuration
Warranty
Distribution
Startup
What can you do with AmigaTalk? Check these out:
General Classes
Intuition Classes
System Classes
User Classes
TheBrowser Browser OverView.
System Directives
System Differences
Appendix 2 (Manual page)

* Smalltalk-80 is a trademark of the Xerox Corporation.

1.2 Unfinished business:

The following parts of the AmigaTalk© program still need work:

1. Verify that TheBrowser is 100% bug-free.

2. Complete program documentation.

3. Write code for loading & saving system image files.

(see)l filename &)s filename)

4. Need to add Class files (System/) for:

Devices

Audio

Narrator

Clipboard

Console

Keyboard

Input

Parallel

Printer

SCSI

Timer

TrackDisk

Processes

Tasks

Memory

Lists

Interrupt

Semaphore

Signal

Exception

ARexx

5. Need to add Class files (Intuition/) for:

AreaPaint

IStruct

Animation

IFF

1.3 User Classes for AmigaTalk© 1998:

The following user classes were written by me (Author) for the AmigaTalk system & are documented herein:

Complex

Curses

Fraction

Please feel free to dream up your own bunch of Classes to add to the system.

1.4 Bugs known to the Author of AmigaTalk:

The following bugs are inherited from the implementation of the Little Smalltalk source code:

1. Assigning a block to a variable will usually cause memory reference cycles, resulting in the number of increments and decrements not being equal following execution.
(not really a bug, caused by the implementation - blocks need to access the context, which includes the local variables, which includes the block, which needs to access the context ...).

2. The stack size allocated by the parser is fixed. It should be computed by the parser and varied for different methods.

3. If you modify the class AmigaTalk (and thereby the pseudo-variable amigatalk) the pseudo-variable does not get properly initialized.

4. Because of bootstrapping peculiarities, a few classes (such as Object or Symbol or String) do not respond to the message variables correctly.

5. The Parser is NOT smart enough to recognize when you've created a conflict with the names of variables. Example:

```
Class Broken :Object
! nameConflict secondConflict !
[
firstMethod: nameConflict ! secondConflict !
...
]
```

Make sure that instance variables & method parameters don't have the same name. The Parser will compile your Class without complaining but there will be a run-time error in your class that won't be easy to find!

1.5 File System Configuration:

This is the layout of the AmigaTalk environment that you need to have

in order to suppress un-wanted requester activity:

Assign AmigaTalk: to a directory or partition: DH0:AmigaTalk/ (example)

AmigaTalk -- The main program.

AmigaTalk.info -- The main program icon.

AmigaTalk.env -- The Screen & Window startup values.

(See DisplayEnvFile)

Sub-directories of AmigaTalk:

Browser/ The user should NOT play with the

GeneralClasses files in this directory, TheBrowser

GeneralClassInstanceVars uses these files & expects a certain

GeneralMethodsInstanceVars structure to them. TheBrowser will

IntuitionClasses maintain them for you.

IntuitionClassInstanceVars

IntuitionMethodsInstanceVars

SystemClasses

SystemClassInstanceVars

SystemMethodsInstanceVars

UserClasses

UserClassInstanceVars

UserMethodsInstanceVars

C/

AmigaTalk.iff (See -i command switch)

CursesHail.st No longer used.

DisplayLogo (See -i command switch)

Parse - Part of the Original Little SmallTalk system.

TheBrowser - CanDo Class-Browsing deck.

TheBrowser.info

CodeLib/ CodeLib/ is Currently empty.

Help/

AmigaTalk.guide - This file.

AmigaTalk.guide.info

ATHelper.guide

GeneralClasses.guide

Intuition.guide

System.guide

SystemDiagram.iff A picture of how the program is connected.

User-supplied (meaning the Author) Classes:

Complex.guide

Curses.guide

Fraction.guide

TheBrowser.guide

ATalkARexx.guide

General/ (Source files for general classes)

AmigaTalk.st

Array.st

ArrayedCollection.st

Bag.st

Block.st

Boolean.st

ByteArray.st

Char.st

Class.st

Collection.st

Dictionary.st

False.st

File.st

Float.st

Form.st

Integer.st

Interval.st

KeyedCollection.st

larray.st

List.st

Magnitude.st

Number.st

Object.st

Pen.st

Point.st

Process.st

Radian.st

Random.st

Semaphore.st

SequenceableCollection.st

Set.st

String.st

Symbol.st

True.st

UndefinedObject.st

Intuition/

intuition.p

Prelude/

standard

System/

system.p

User/

Complex.st

Curses.st

Fraction.st

user.p

1.6 AmigaTalk requirements:

REQUIREMENTS

~~~~~~~~~~~~

AmigaTalk uses DeckRunner which is Copyrighted (c) by InovaTronics.
I've included a copy of it with my distribution. It should be placed
in the command path where the Amiga OS can find it (example: C:)
It also requires the cando.library & CandoGFX.library. Let me know if
you have trouble obtaining a copy of these.
68020 or better microprocessor.
AmigaDOS 3.1+
Hard disk.

1.7 NO warranty for AmigaTalk:

NO WARRANTY

~~~~~~~~~~~

There is no warranty for this software package. Although the Author
has tried to prevent errors, he can't guarantee that the software
package described in this document is 100% reliable. In fact, I can
certainly guarantee that there are some problems in the code. You are
therefore using this material at your own risk. The author cannot be
held responsible for any damage which is caused by using this software
package. Hey, what do you want for free?
(See Disclaimers)
I would appreciate any bug reports & I would suggest that you send them
via e-mail to my attention: jsteic1957@aol.com

1.8 If it don't work, you're to blame:

You get to figure out which ones apply to this program ;)

No other warranty expressed or implied.

List was current at time of printing.

This supersedes all previous notices.

This information is subject to change without notice.

All rights reserved.

Do not purchase if seal has been tampered with.

Contents under pressure. Do NOT incinerate.

Some assembly required.

List each check separately by bank number.

Batteries not included.

Some settling of contents may occur during shipping.

Use only as directed.

Do not use while operating a motor vehicle or heavy equipment.

This product is meant for educational purposes only.

Postage will be paid by addressee.

This is not an offer to sell securities.

Apply only to affected area.

May be too intense for some viewers.

Do not stamp.

For recreational use only.

All models over 18 years of age.

If condition persists, consult your physician.

No user-serviceable parts inside.

Times approximate.

Simulated picture.

Breaking seal constitutes acceptance of agreement.

One size fits all.

Contains a substantial amount of non-tobacco ingredients.

Colors may, in time, fade.

We have sent the forms which seem to be right for you.

Slippery when wet.

Parental guidance suggested.

Do not bend, fold, spindle, or mutilate.

We reserve the right to refuse service to anyone.

Use unleaded fuel only.

Not affiliated with the American Red Cross.

Drop in any mailbox.

Edited for television.

Keep cool; process promptly.

Post office will not deliver without postage.

Not responsible for direct, indirect, incidental or consequential damages resulting from any defect, error or failure to perform.

No part of this publication may be reproduced in any form or by any means without the prior written permission of the publisher.

Some of the trademarks mentioned in this product appear for identification purposes only.

At participating locations only.

Your mileage may vary.

Substantial penalty for early withdrawal.

Lost ticket pays maximum rate.

Your cancelled check is your receipt.

Avoid contact with skin.

Be sure each item is properly endorsed.

Sign here without admitting guilt.

Other items sold separately.

Slightly higher in California.

Employees and their families are not eligible.

Limited time offer, call now to insure prompt delivery.

No passes accepted for this engagement.

Use only in well-ventilated area.

Keep away from fire or flame.

Replace with same type.

Approved for veterans.

Booths for two or more.

Some equipment shown is optional.

Don't try this at home.

Dispose of properly.

Prerecorded for this time zone.

Reproduction strictly prohibited.

Do not immerse.

No solicitors.

No alcohol, dogs, or horses.

List at least two alternate dates.

Record additional transactions on back of previous stub.

Price does not include taxes, title, or destination charges.

This is not a competition, it is only an exhibition.

No wagering.

1.9 AmigaTalk distribution:

DISTRIBUTION

~~~~~~~~~~~~

This software package is ShareWare. It may be put on any
media which is used for the distribution of ShareWare, like Public
Domain disk collections, CDROMs or FTP servers.
In order to ensure the integrity of this software package, distributors
should use the complete original distribution. The author cannot be held
responsible if this software has become unusable due to modifications
of the distribution contents or of the distribution files themselves.
There is no limit on the costs of the distribution, e.g. for the media,
like floppy disks, streamer tapes or compact disks, or the process of
duplicating. Such limits have been proven to be harmful to the idea of
shareware, i.e., instead of reducing the price of the
floppy disk below the limit, the software was simply removed from the
master disk.
Although the Author does not impose any limit on the
distribution of this software, he would like to express his personal
opinions on this matter:
* This software package should be made available to everyone free of
charge whenever it is possible (so they can send me money instead ;)).
* If you have acquired this software package under normal conditions
from a Public Domain dealer on a floppy disk at a price higher than
US $5, then you have definitely paid too much. Please don't
support this improper profit making any longer and switch to a
less expensive source as soon as possible.

1.10 Running the system:

The AmigaTalk system is invoked by typing the command AmigaTalk
or by clicking on the AmigaTalk icon. The system is interactive - that
is, the user types an expression at the keyboard and the system responds
by evaluating the expression and typing the result. For example, typing
the expression 3 + 4 results in the value 7 being displayed in the
Status Window. Execution is terminated by selecting Quit
from the Menu. The program will open its own screen & windows.
Instance variables for the command level can be created by assigning a
value to a new variable name. Thereafter that variable can be used at
~~~~~~~~~~~~

the command level, although it is not known within the scope of any
method loaded in the system. The variable last always contains
the value returned by the last expression typed. Figure 2 shows the
creation of a variable. Note that the assignment arrow is formed as a
two-character sequence from < and -:

```
newvar <- 2 / 3 (stuff you type in)
newvar
0.666667 (result displayed in Status Window)
2 raisedTo: newvar + (4 / 3)
4
last
4
```

Figure 2: Creating Variables

The default behavior is for the value of expressions, with the exception
of assignments, to be typed automatically as they are evaluated. This
behavior can be modified either by using the -d flag (see
Appendix 2), or by passing a message to the pseudo-variable
amigatalk (see Appendix 1).

See Also System Directives

System Configuration

1.11 System Directives:

Class descriptions must be read in from files, they cannot be entered
interactively in the GUI. Pressing the Parse button will
bring up a file requester so that the user can select a Class source file
for inclusion into the internal Class Dictionary of AmigaTalk. You can
also load a class description contained in a file named newclass.st
by typing the following system directive in the Command Line string
Gadget:

```
)i newclass.st
```

A list of files containing class descriptions can also be given as
arguments to the AmigaTalk command (CLI only!). The command

```
AmigaTalk file1 ... filen
```

is equivalent to the sequence

```
)i file1
...
)i filen
```

being typed into the Command Line Gadget.

See Also,

)e filename Invoke the editor.

)g filename Parse & Load a Class.

)i filename Parse & Load a Class.

)r filename Read Commands file.

)l filename Load Saved Image.

)s filename Save System Image.

)! command Execute a CLI command.

1.12 Parse Button Behavior:

The Parse button will ask for the file you want to parse using the
ubiquitous ASL file requester. It then issues a)i system directive
so that AmigaTalk will include the file class contents in memory. The
file will be parsed & added to the AmigaTalk internal Class dictionary.
Be sure that the file has already been debugged, since the parser has
no way of returning error notices to the AmigaTalk screen. You can do
this by typing a command into a Shell or CLI:

AmigaTalk:c/Parse <myClass.st >myClass.p

The parser will send error strings back to the Shell or CLI.

1.13 Format for Env file:

1.14 AmigaTalk icon ToolTypes:

The following ToolTypes can be specified in the AmigaTalk icon:

ToolType: Default Values:

DisplayEnvFile =AmigaTalk:AmigaTalk.env (See File Format).

Editor =C:Ed

DefaultEnvFile =AmigaTalk:Prelude/standard

LibraryPath =AmigaTalk:CodeLib/

ParserName =AmigaTalk:c/Parse

CommandPath =[AmigaTalk:c/]

HelpPath =[AmigaTalk:Help/]

HelpProgram =MultiView

GeneralPath =[AmigaTalk:General/]

IntuitionPath =[AmigaTalk:Intuition/]

SystemPath =[AmigaTalk:System/]

UserPath =[AmigaTalk:User/]

ARexxPortName =[AmigaTalk_Rexx]

DefaultTabSize =[3] (Not currently used).

StatusHistoryLength =[20] (Not currently used).

ColorsFile =[AmigaTalk:AmigaTalk.colors]

LogoName =AmigaTalk.iff

LogoCmd =DisplayLogo

See Also, TheBrowser ToolTypes

1.15 System Directive)e (Editor):

Note that the)e system directive invokes an editor on a file containing class descriptions, and then automatically includes the file when the editor is exited. Classes also respond to the message edit, which will have the same effect as the)e directive applied to the file containing the class description. Thus the typical debug/edit/debug cycle involves repeated uses of the)e directive or the edit message until a desired outcome is achieved. The editor invoked by the)e directive can be changed by setting the EDITOR= ToolType in the AmigaTalk icon.

)e filename

Edit the named file. The AmigaTalk system will suspend, leaving the user in an editor for making changes to the named file. Upon leaving the editor, the named file will automatically be included, as if the)i system directive had been typed.

1.16 System Directive)g (Read):

NOTE: This system directive command is similar to)r, except that it searches AmigaTalk:CodeLib/ for the file to read in.

)g filename

Search for an entry in the system library area matching the filename. If found, the class descriptions in the library entry are included. This command is useful for including commonly used classes that are not part of the standard prelude, such as classes for statistics applications or graphics.

See Also)r (Read)

1.17 System Directive)i (Include class):

)i filename

Include the named file. The file must contain one or more class
descriptions. The class descriptions are parsed, and if syntactically
legal, new instances of the classes are added to the AmigaTalk system.
WARNING: No error messages from the parser will be returned to you.
you must parse the class file from a Shell or CLI, in order
to make sure that it's syntactically legal.

1.18 System Directive)l (Load):

WARNING: This command is NOT currently implemented!

)l filename

Load a previously saved environment from the named file. The current
values of all variables are overridden. The file must have been created
using the)s directive.

1.19 System Directive)r (Read):

NOTE: This system directive command is similar to)g, except that
it does NOT search AmigaTalk:CodeLib/ for the file to read in.

)r filename

Read the named file. The file must contain AmigaTalk statements, as
they would be typed at the keyboard. The effect is just as if the
lines of the file had been typed at the keyboard. The file cannot
contain class descriptions.

NOTE: This operation can also be performed by selecting the Load menu
item from the menu.

See Also)g (Read)

1.20 System Directive)s (Save):

WARNING: This command is NOT currently implemented!

)s filename

Save the current state in the named file. The values of all variables
are saved, and can later be reloaded using the)l directive.

1.21 System Directive)! (Shell):

)!string
Execute the remainder of the line following the exclamation point as an AmigaDOS command. Nothing is done with the output of the command, nor is the returning status of the command recorded.

1.22 System Differences:

Differences between AmigaTalk & the Smalltalk-80 system:
This section describes the differences between the language accepted by the AmigaTalk system and the language described in Smalltalk blue.
The principal reasons for these changes are as follows:
size
Classes which are largely unnecessary, or which could be easily simulated by other classes (e.g. Association, SortedCollection) have been eliminated in the interest of keeping the size of the standard library as small as possible. Similarly, indexed instance variables are not supported, since to do so would increase the size of every object in the system, and they can be easily simulated in those classes in which they are important (see below).
representation
The need for a textual representation for class descriptions required some small additions to the syntax for class methods (see Appendix 3). Similarly, the fact that classes and subclasses can be separately parsed, in either order, forced some changes in the scoping rules for instance variables.
The following sections describe these changes in more detail:
3.1. Internal Representation Different
The internal representations of objects, including processes, interpreters, and bytecodes, is entirely different in the AmigaTalk system from the Smalltalk-80 system described in AmigaTalk blue.
If you read A Little Smalltalk, AmigaTalk is pretty much the same system, except that there are more defined primitives, allowing the full use of the Amiga Operating System.
3.2. FewerClasses
Many of the classes described in Smalltalk blue are not included as part of the AmigaTalk basic system. Some of these are not necessary because of the interest in keeping the standard library of classes small.

A complete list of included classes for the AmigaTalk system is given in Appendix 1 .

3.3. No Class Protocol

Protocol for all classes is defined as part of class Class. It is not possible to redefine class protocol as part of a class description, only instance protocol. The notion of metaclasses is not supported.

3.4. Cascades Different

The semantics of cascades has been simplified and generalized. The result of a cascaded expression is always the result of the expression to the left of the first semicolon, which is also the receiver for each subsequent continuation. Continuations can include multiple messages. A rather non-sensical, but illustrative, example is the following:

```
2 + 3 ; - 7 + 3 ; * 4
```

The result of this expression is 5 (the value yielded by 2 + 3). 5 is also the receiver for the message - 7, and that result (-2) is in turn the receiver for the message + 3. This last result is thrown away. 5 is then again used as the receiver for the message * 4, the result of which is also thrown away.

3.5. Instance Variable Name Scope

In the language described in Smalltalk blue, an instance variable is known not only to the class protocol in which it is declared, but is also valid in methods defined for any subclasses of that class. In the AmigaTalk system an instance variable can be referenced only within the protocol for the class in which it is declared.

3.6. Indexed Instance Variables

Implicitly defined indexed instance variables are not supported. In any class for which these are desired they can be easily simulated by including an additional instance variable, containing an Array, and including the following methods:

```
Class Whatever
! indexVars !
[
new: size
indexVars <- Array new: size
|
at: location
^ indexVars at: location
|
at: location put: value
```

indexVars at: location put: value

...

]

The message new: can be used with any class, with an effect similar to new. That is, if a new instance of the class is created by sending the message new: to the class variable, the message is immediately passed on to the new instance, and the result returned is used as the result of the creation message.

3.7. No Pool Variables

The concepts of pool variables, global variables, or class variables are not supported. In their place there is a new pseudo-variable, amigatalk, which responds to the messages at: and at:put:. The keys for this collection can be arbitrary. Although this facility is available, its use is often a sign of poor program design, and should be avoided.

3.8. No Associations

The class Dictionary stores keys and values separately, rather than as instances of Association. The class Association, and all messages referring to Associations have been removed. Creating an Association Class is left as an exercise for the user ;)

3.9. Generators in place of Streams

The notion of stream has been replaced by the slightly different notion of generators, in particular the use of the messages first and next in subclasses of Collection. External files are supported by an explicit class File.

3.10. Primitives Different

Both the syntax and the use of primitives has been changed. Primitives provide an interface between the AmigaTalk world and the underlying Amiga operating system, permitting the execution of operations that cannot be specified in AmigaTalk. In AmigaTalk, primitives cannot fail and must return a value (although they may, in error situations, print an error message and return nil). The syntax for primitives has been altered to permit the specification of primitives with an arbitrary number of arguments. The format for a primitive call is as follows:

<primitive number argumentlist>

Where "number" is the number of the primitive to be executed (which must be a value between 1 and 255, 0 does nothing), and "argumentlist" is a list of AmigaTalk primary expressions (see Appendix 2). Appendix 4 lists the meanings of each of the currently recognized primitive numbers.

3.11. Byte Arrays

A new syntax has been created for defining an array composed entirely of unsigned integers in the range 0-255. These arrays are given a very tight encoding. The syntax is a pound sign, followed by a left square brace, followed by a sequence of numbers in the range 0 to 255, followed by a right square brace.

#[numbers]

Byte Arrays are used extensively inside the AmigaTalk system.

3.12. New Pseudo-Variables

In addition to the pseudo-variable amigatalk already mentioned, another pseudo-variable, selfProcess, has been added to the AmigaTalk system. selfProcess returns the currently executing process, which can then be passed as an argument to a semaphore, or be used as a receiver for a message valid for class Process. Like self and super, selfProcess cannot be used at the command level.

3.13. No Dependency

The notion of dependency, and automatic dependency updating, is not included in AmigaTalk.

1.23 Appendix 1 - Class Descriptions:

The messages accepted by the classes included in the AmigaTalk standard library are described in the following pages. A list of the classes defined, where indentation is used to imply subclassing, is given below:

Object
UndefinedObject
Symbol
Boolean
True
False
Magnitude
Char
Number
Integer
Float
Radian
Point
Random
Collection

Bag

Set

KeyedCollection

Dictionary

AmigaTalk

File

SequenceableCollection

Interval

LinkedList

Semaphore

File

ArrayedCollection

Array

ByteArray

String

Block

Class

Process

Pen

In the descriptions of each message the following notes may occur:

d Indicates the effect of the message differs slightly from that
given in Smalltalk blue.

n Indicates the message is not included as part of the language
defined in Smalltalk blue.

r Indicates the protocol for the message overrides a protocol given
in some superclass. Only where the logical effect of this over-
riding is important is the message given a second time; some
messages, such as copy, are overridden in many classes but are not
described in the documentation because the logical effect remains
the same.

1.24 Appendix 2 - Manual page:

AmigaTalk(1) (local) AmigaTalk(1)

NAME

AmigaTalk - a derivative of Little Smalltalk

SYNOPSIS

AmigaTalk [options] [files]

DESCRIPTION

AmigaTalk is an interpreter for a Smalltalk-like language.
For a complete description of the language accepted by the
interpreter see the user manual. Options accepted on the
command line (CLI) are as follows:
-h Print usage information.
-i Display Logo window (Default = OFF).
-a If the -a option is given statistics on the number of
memory allocations will be displayed following
execution.
-d digit
If the "digit" is zero only those results explicitly
requested by the user will be printed. If 1, the
values of expressions typed at the keyboard will be
displayed (this is the default). If 2, the values of
expressions and the values assigned in assignment
statements will be displayed.
-g The next argument is taken to be the name of an
additional library stored in the system library area
(see LibraryPath ToolType).
The library is loaded following the standard prelude,
just as if a)g directive were given at the
beginning of execution.
-n The -n option, if given, suppresses the loading of the
standard library. As this gives you a system with
almost no functionality, it is seldom useful except
during debugging.
-z print lexical analyzer debugging statements.
-b No Status Window (Default = ON).
-r The next argument is taken to be the name of a file of
Smalltalk commands. The file is included prior to
execution, just as if a)r directive were given at
the beginning of execution.
-s In normal operation, at the end of execution the number
of reference count increments and decrements is printed
just prior to exit. In the absence of cycles these two
figures should be equal. Since cycles can cause large
chunks of memory to become unreachable, and seriously
degrade performance, this information is often useful
in debugging. The -s option, if given, suppresses the

printing of this information.

The following options are best left untried, since fast loading
and reading & saving of binary images of the internal system
state are not currently implemented:

-l The next argument is taken to be the name of a file
containing a binary image saved using the)s directive
(see below). This binary image is loaded prior to
execution.

-m Do not perform fast loading. (Used when fastloading is
the default).

-f The -f option indicates fast loading should be used,
which loads a binary save image (see ``)s” below) for
the standard library.

The files, if given, must contain class descriptions.
Consult the reference manual for the syntax for class descriptions.
The classes defined are included along with the standard library of
classes before execution begins.

Once execution begins, the Status Window will display

Welcome to AmigaTalk!

indicating that the system is ready for a command to be entered.
A command consists of a valid Smalltalk expression, without a
terminating period. As each expression is entered it is executed
by the Smalltalk interpreter, and the results displayed.

The following system directives can be entered in place of
commands:

)e filename

Edit the named file, which must contain only class descriptions.
The AmigaTalk system will suspend, leaving the user in an editor
for making changes to the named file. Upon exiting the editor,
the named file will automatically be included, as with the)i
directive (below). The editor chosen by this command is taken
from the Editor ToolType in the AmigaTalk icon.

)g filename

Search for a file with the given name in the system library area.
If found, load the library in with the users classes. This is
useful for creating libraries of commonly used classes which are
not part of the standard prelude, such as classes for statistics
applications or for graphics.

)i filename

Include the named file. The File must contain one or more class
descriptions. The class descriptions are parsed, and if syntact-
ically legal, new instances of the classes are added to the
AmigaTalk system.
)r filename
Read the named file. The effect is just as if the lines in the
file had been typed at the keyboard. The file cannot contain
class descriptions.
)!string
Execute the string following the exclamation point as a
system command.
The following options are best left untried, since reading &
saving of binary images of the internal system state are not
currently implemented:
)l filename
Load a saved binary environment. The file must have been prev-
iously created using the)s directive (below). The values of all
variables are overridden.
)s filename
Save the current environment in the named file. The values of all
variables will be saved, and can later be restored using the
)l directive (above).
Author
James T. Steichen, for the Amiga-specific code &
Tim Budd, Department of Computer Science, The University of
Arizona, for the original Little Smalltalk programming system.
See Also
Timothy A. Budd, "A Little Smalltalk Users Manual" &
"A Little Smalltalk" by Timothy A. Budd, Addison Wesley, 1987
Bugs
Not all the Smalltalk-80 Language described in the Blue Book
is supported; see the user manual for details. (Smalltalk-
80 is a trademark of Xerox Corporation).

1.25 Appendix 3:

The following is an example class description:
Class Set :Collection
! dict !

```
[
new
dict <- Dictionary new
|
add: newElement
dict at: newElement
ifAbsent: [dict at: newElement put: 1]
|
remove: oldElement ifAbsent: exceptionBlock
dict removeKey: oldElement ifAbsent: exceptionBlock
|
size
^ dict size
|
occurrencesOf: anElement
^ dict at: anElement ifAbsent: [0]
|
first
dict first.
^ dict currentKey
|
next
dict next.
^ dict currentKey
]
```

1.26 Appendix 4 - Primitive Numbers:

The following list gives the function performed by each primitive in
the AmigaTalk system.

Information about objects:

0 (not used)

1 class of an object

2 superobject of an object

3 test if class responds to new

4 size of object

5 hash value

6 test if two built-in objects are of the same type

7 object equality testing (==)

8 various switch toggles

9 numerical generality testing

Integer Manipulation:

10 integer addition (both args must be integer)

11 integer subtraction

12 integer < test

13 integer > test

14 integer <_ test

15 integer >_ test

16 integer = test

17 integer ~= test

18 integer multiplication

19 integer //

Bit Manipulation & other integer-valued functions:

20 gcd:

21 bitAt:

22 bitOr:

23 bitAnd:

24 bitXor:

25 bitShift:

26 radix:

27 not used

28 integer quo:

29 integer rem:

Other integer functions:

30 doPrimitive:withArguments:

31 not used

32 convert random integer to random float

33 bitInvert

34 highBit

35 randomNumber (argument is seed)

36 asCharacter

37 asString

38 factorial

39 asFloat

Character manipulation:

40 not used

41. not used

42 character < test

43 character > test

44 character <_ test

45 character >_ test

46 character = test

47 character ~= test

48 not used

49 not used

Character unary functions:

50 digitValue

51 isVowel

52 isLetter

53 isLowerCase

54 isUpperCase

55 isSeparator

56 isAlphaNumeric

57 caseShift

58 asString

59 asciiValue

Floating point manipulation:

60 floating point addition (both args must be float)

61 floating point subtraction

62 floating point < test

63 floating point > test

64 floating point <_ test

65 floating point >_ test

66 floating point = test

67 floating point ~= test

68 floating point multiplication

69 floating point division

Other floating point operations:

70 ln

71 sqrt

72 floor

73 ceiling

74 not used

75 integerPart

76 fractionalPart

77 gamma

78 asString

79 exp

Other numerical functions:

80 normalize number to be within 0 and 2 * pi.

81 sin

82 cos

83 not used

84 arcSin

85 arcCos

86 arcTan

87 not used

88 raisedTo:

89 radix:

Symbol Commands:

90. not used

91 symbol comparison, returns true or false.

92 printString

93 asString

94 print (used internally)

95 not used

96 not used

97 build a new class, arguments are class name, superclass

name, source filename, instance variables, messages, methods,

context size & maximum stack size (eight arguments!).

98 insert an object into class dictionary, first argument

is symbol, second argument is class definition

99 find an object in class dictionary. argument is symbol.

String operations:

100 string length

101 string compare, case important - return -1, 0 or 1.

102 string compare, case not important

103 string catenation

104 string at:

105 string at:put:

106 copyFrom:length:

107 copy (new string with same chars)

108 asSymbol

109 string printString

Array manipulation:

110 build an untyped object of given size, argument is

integer size.

111 index variable get (first argument is object, second is index)

112 index variable put (first argument is object, second is index, third argument is expression)

113 object grow (returns a new object with same instance variable values as first argument, but with second argument tacked on end as new instance variable)

114 build an instance of "Array" of the given size.

115 new string of given size

116 ByteArray new:

117 ByteArray size

118 ByteArray at:

119 ByteArray at:put:

Output & error messages:

120 print string with no return

121 print string with return

122 general error - first argument is receiver, second is error string.

123 print string on error output (with return)

124 Curses library interface primitives.

See Curses interface for more information.

125 unix system call

126 print a string at a specific point on the terminal.

See Curses interface for more information.

127 block return without surrounding context

128 reference count less than zero, first argument is guilty object.

129 does not respond error, first argument is receiver, second is message.

File operations:

130 file open, first argument is name, second argument is mode

131 file read

132 file write

133 set file mode, first argument is file, second is mode indicator (anInteger)

134 compute file size in bytes

135 file set location (at:) second argument is location

(anInteger)

136 return current file offset in bytes

137 not used

138 not used

139 not used

Process management:

NOTE: This is NOT the same as Exec Processes that are utilized by the

Amiga OS.

140 block execute (trapped by interpreter)

141 new process (withArguments:)

142 terminate a process

143 perform:withArguments: (trapped by interpreter)

144. not used

145 set state

146 return state

148 start atomic action

149 end atomic action

Operations on classes:

150 class edit

151 superclass of a class

152 class name (a Symbol)

153 new instance of a class

154 list all commands class responds to

155 respondsTo: , second argument is a symbol

156 class view (drop into editor, but no include)

157 class list

158 variables (returns an array of symbols)

159 get ByteCode representation of method.

Date & Time:

160 current date and time as string

161 seconds time counter

162 clear the screen (Curses function)

Plot(3) interface: (Only 20 (max) Allowed at one time!)

NOTE: These primitives have been re-worked so that they are

Amiga-specific. So DON'T expect any code you write to be

portable!

169 Open, CLose or Move a PlotEnv Window.

170 clear the Plot window.

171 move the Plotting Pen.

172 draw a line from the current location to the given location.
173 draw a point at the given location.
174 draw a circle.
175 draw a box (Box(x0, y0, x1, y1))
176 set the drawing pens to Color registers. (pens(front, back))
177 draw a line (line(x1, y1, x2, y2))
178 print a label (label(string, x, y))
179 establish a line type (SetDrPt(bitPattern))
Primitives 180 through 255 are purposely NOT documented here, since
they are used to provide the functionality of the Amiga OS to the
AmigaTalk system. You can probably figure out what they are from
the Smalltalk files in the Intuiion/ & System/ directories if you
are determined to pirate the knowledge; but I have a better idea for
you - why not re-write & improve the Classes as I've defined them?

1.27 Author Information:

If your conscience is bugging you to contribute some cash for my
programming efforts, please send $5 or more for the program to:
The Author:
James T. Steichen (A real cool frood!)
2217 N. Tamarack Dr.
Slayton, Mn. 56172-1155 (USA)
email: jsteic1957@aol.com (best place to send bug reports)
AmigaTalk was written in C from the Original Source code for
Little Smalltalk V1.0 using the SAS C compiler V6.58.
TheBrowser was written using CanDo! V3.008 by InovaTronics.

1.28 Bibliography:

This is only a partial list of books available on Smalltalk &
Little Smalltalk:
A Little Smalltalk by Timothy Budd, Addison Wesley, 1987
ISBN: 0-201-10698-1
Smalltalk-80, The Language by Adele Goldberg & David Robson,
Addison Wesley, 1989 (the Purple book)
ISBN: 0-201-13688-0
Smalltalk-80, The Language & its Implementation
by Adele Goldberg & David Robson, (the Blue book).
Addison Wesley, 1983
Smalltalk-80, The Interactive Programming Environment
by Adele Goldberg & David Robson, (the Orange book).
Addison Wesley, 1983